SEGA™

SEGA OF AMERICA, INC.
Consumer Products Division

32X
Technical Information
Attachment 1

Doc. # MAR-42-072694

32X Technical Information Attachment 1

Details of 32X Technical Information 6

When using the 32X, and the RV bit of A15106H addess is "1", the normal operation of the Mega Drive can be affected after reset is applied. To correct this, the hardware has been changed so that the 32X system is reset by the watch-dog-timer output when VRES interrupt occurs on the SH2 (Master) side, and the RV bit is checked and is "1".

With respect to each application, the determination must be made whether or not the SH2 resets the system by checking the RV bit in the process within the VRES interrupt. On the MD side, the initial program operates if the system is reset, but because the MD side I/O isn't reset, the initial program moves onto application execution without executing the adapter usage procedure and determines whether or not the adapter usage procedure is performed; if the procedure hasn't been performed, it then must be performed .

Apart from the above procedure, it must be determined whether all processes at the start time are performed as a corrective measure when reset is applied repeatedly. With regard to applications that don't change the RV bit, the above operation is not required.

The above corrective measure will go into effect from the Ver. 2.1 (new board scheduled for release after Sept. 1994) development board. This problem cannot be avoided for development boards prior to Ver. 2.1 even if corrective action is taken by software.

The corrective measure with respect to the actual program is shown below. (From the sample program).

68000 Side Corrective Program Sample

```
*-------------------------------------------------------------*
; Vector / Mega Drive ID / Mars Initial Program
        .include    source¥header.prg       ; Mega Drive & Mars Header
        .include    source¥icd_mars.prg     ; Sega indicated Initial Program & Security
*-------------------------------------------------------------*

        bcs         _error0                 ; if cs = 1  then ID error
                                            ;    or Self check error

        move.l      #0, initflug            ; clear initial flag
        btst        #15, d0
        bne.b       VresStart               ; Reset with VRES Button?

        bra         _init
```

```
VresStart:
                lea     marsreg, a5
                btst.b  #ADEN, adapter (a5)
                bne     AdapterEnable           ; has 32X gone into effect?

*----           SUPER 32X Usage Procedure

                move.l  #0, comm8 (a5)

                lea     ?10, a0                 ; copy from ROM to WRAM
                lea     $ff0000, al

                move.l  (a0)+, (a1)+
                move.l  (a0)+, (a1)+
                move.l  (a0)+, (a1)+
                move.l  (a0)+, (a1)+
                move.l  (a0)+, (a1)+
                move.l  (a0)+, (a1)+
                move.l  (a0)+, (a1)+

                lea     $ff0000, a0
                jmp     (a0)                    ; jump workram
?10:
                move.b  #1, adapter (a5)        ; SUPER 32X Mode
                                                ; SH2 reset - wait 10ms -
                lea     Restarticd, a0
                adda.l  #marsipl, a0
                jmp     (a0)                    ; jump ROM (+$880000)
Restarticd:
                lea     $a10000, a5
                move.l  #-64, a4
                move.w  #3900, d7               ; 8
                lea     marsipl+$6e4, a1
                jmp     (a1)                    ; jump icd_mars.prg ?res_wait
AdapterEnable:
                lea     marsreg, a5
                btst.b  #RES, adapter (a5)
                bne     _hotstart               ; SH2 reset canceled?
                bra.b   Restarticd              ; If not canceled reset once again
                                                ; operate icd_mars.prg
*-------------------------------------------------------*
*       Main Program
*-------------------------------------------------------*

_init:
                lea     marsreg, a5
        ?w:
                cmp.l   #'M_OK' , comm0 (a5)    ; SH2 Master OK ?
                bne.b   ?w
        ?w1:
                cmp.l   #'S_OK' , comm4 (a5)    ; SH2 Slave OK ?
                bne.b   ?w1

                moveq   #0, d0                  ; SH2 Start
                move.l  d0, comm0 (a5)          ; Master
```

```
                move.l      d0, comm4 (a5)          ; Slave
                move.l      #"INIT", initflug
_hotstart:
                cmp.l       #"INIT", initflug       ; Has initial process ended ?
                bne.b       _init
                move.l      $880000, a7
                bsr         SysInit                 ; Mega Drive Init
        ?start:
                move.w      #$2000, sr
                move.w      #$8164, reg_1 (a6)      ; Display On
                move.w      #$8164, _vdpreg         ; V Interrupt Enable
```

SH2 (Master) Side Corrective Program Sample

```
;-------------------------------------------------------*
;       SH2 (Master) Vector
;-------------------------------------------------------*

vector:
        .data.l         start                   ; Power On Reset PC
_stack:
        .data.l         M_STACK,                ; Power On Reset SP
+                       start,                  ; Manual Reset PC
+                       M_STACK                 ; Manual Reset SP

        .data.l         error0,                 ; General invalid command
+                       h'00000000,             ; System reserve
+                       error0,                 ; Slot invalid command
+                       h'20100400,             ; System reserve  (ICE Vector)
+                       h'20100420,             ; System reserve  (ICE Vector)
+                       error0,                 ; CPU address error
+                       error0,                 ; DMA address error
+                       error0,                 ; NMI
+                       error0                  ; User break

        .datab.l        19, h'00000000          ; System reserve
        .datab.l        32, error0              ; Trap command (User vector)

        .data.l         m_int,                  ; Interrupt 1
+                       m_int,                  ; Interrupt 2, 3
+                       m_int,                  ; Interrupt 4, 5
+                       m_int,                  ; Interrupt 6, 7
+                       m_int,                  ; Interrupt 8, 9
+                       m_int,                  ; Interrupt 10, 11
+                       m_int,                  ; Interrupt 12, 13
+                       m_int                   ; Interrupt 14, 15

;-------------------------------------------------------*
;       Program Start
;-------------------------------------------------------*

start:
                mov.l       #_sysreg, r14
                ldc         r14, gbr

                mov.l       #_FRT, r1               ; Set Free Run Timer
                mov         #h'00, r0
```

```
                mov.b   r0, @ (_TIER, r1)       ; for Correcting Interrupt
                mov     #h'e2, r0
                mov.b   r0, @ (_T0CR, r1)       ;
                mov     #h'00, r0
                mov.b   r0, @ (_0CR_H, r1)      ;
                mov     #h  01, r0
                mov.b   r0, @ (_0CR_L, r1)      ;
                mov     #0, r0
                mov.b   r0, @ (_TCR, r1)        ;
                mov     #1  r0
                mov.b   r0, @ (_TCSR, r1)       ;
                mov     #h'  00, r0
                mov.b   r0, @ (_FRC_L, r1)      ;
                mov.b   r0, @ (_FRC_H, r1)      ;

                mov     #h'_f2, r0              ; for Correcting VRES
                mov.b   r0,@ (_T0CR, r1)        ;
                mov     #h  00, r0
                mov.b   r0, @ (_0CR_H, r1)      ;
                mov     #h  01, r0
                mov.b   r0, @ (_0CR_L, r1)      ;
                mov     #h  e2, r0
                mov.b   r0, @ (_T0CR, r1)       ;
wait_md:
                mov.l   @ (comm0, gbr), r0      ; Combine Mega Drive
                                                ; and timing

                cmp/eq  #0, r0
                bf      wait_md

                mov.l   #"SLAV",r1
wait_slave:
                mov.l   @ (comm8, gbr), r0      ; Combine SH2 Slave
                                                ; and timing

                cmp/eq  r1, r0
                bf      wait_slave

                mov.l   #_SERIAL, r1
                mov     #b' 10000000, r0
                mov.b   r0, @r1                 ; Serial Mode Register
                mov     #74, r0
                mov.b   r0, @ (1, r1)           ; Bit Rate Register
                mov     #b' 00000000, r0
                mov.b   r0, @ (2, r1)           ; Serial Control Register
                mov.l   #4*74, r0
w_serial:
                nop
                dt      r0
                bf      w_serial
                mov     #b' 00100000, r0
                mov.b   r0, @ (2, r1)           ; Serial Control Register (start)
                mov     #0, r0
                mov.b   r0, @ (4, r1)
_hotstart:
                mov     #h' 20, r0
                ldc     r0, sr                  ; SH2 Interrupt Enable
                        |
                        |
                        |
                        |
```

```
;-------------------------------------------------------*
;       Interrupt Control
;-------------------------------------------------------*

m_int:
                push    0, 1
                sts.l   pr, @ - r15

                stc     sr, r0
                shlr2   r0
                and     #h' 3c, r0
                mov.l   #inttable, r1

                add     r1, r0
                mov.l   @r0, r1
                jsr     @r1
                nop

                lds.l   @r15+, pr
                pop     0, 1
                rte
                nop

                .align          4
inttable:
        .data.l         noret,          ; Illegal Interrupt
+               noret,          ; Level 1
+               noret,          ; Level 2
+               noret,          ; Level 3
+               noret,          ; Level 4
+               noret,          ; Level 5
+               pwmint,         ; Level 6
+               pwmint,         ; Level 7
+               cmdint,         ; Level 8
+               cmdint,         ; Level 9
+               hint,           ; Level 10
+               hint,           ; Level 11
+               vint,           ; Level 12
+               vint,           ; Level 13
+               vresint,        ; Level 14
+               vresint,        ; Level 15

noret:
                rts
                nop

;-------------------------------------------------------*
;       VRES Interrupt
;-------------------------------------------------------*

vresint:
                mov.l   #_sysreg, r0
                ldc     r0, gbr

                mov.w   r0, @ (vresintclr, gbr)     ; V Interrupt Clear

                mov.b   @ (dreqctl, gbr), r0        ; VRES corrective action
                tst     #RV, r0
                bf      mars_reset
```

```
                mov.l   #M_STACK - 8, r15       ; Stack change
                mov.l   #_hotstart, r0
                mov     r0, @r15                ; PC change
                mov.w   #h f0, r0
                mov     r0, @ (4, r15)          ; SR mask

                mov.l   #_DMAOPERATION, r1
                mov     #0, r0
                mov.l   r0, @r1                 ; DMA Off

                mov.l   #_DMACHANNEL0, r1
                mov     #0, r0
                mov.l   r0, @r1

                mov.l   #b'0100010011100000, r1
                mov.l   r0, @r1                 ; Channel Control

                rte
                nop

mars_reset
                mov.l   #_FRT, r1               ; System Reset
                mov.b   @ (_T0CR, r1), r0       ;
                or      #h' 01, r0              ;
                mov.b   r0, @ (_T0CR, r1)       ;
vresloop:
                bra     vresloop

                        |
                        |
```

Corrective Program Sample of SH2 (Slave) Side

```
                        |
                        |
;-------------------------------------------------------*
;       Interrupt Control
;-------------------------------------------------------*

s_int:
                push    0, 1
                sts.l   pr, @-r15

                stc     sr, r0
                shlr2   r0
                and     #h' 3c, r0
                mov.l   #inttable, r1
                add     r1, r0
                mov.l   @r0, r1
                jsr     @r1
                nop

                lds.l   @r15+ pr
                pop     0, 1
                rte
                nop

                .align  4
```


```
inttable:
        .data.l     noret,          ; Illegal Interrupt
+                   noret,          ; Level 1
+                   noret,          ; Level 2
+                   noret,          ; Level 3
+                   noret,          ; Level 4
+                   noret,          ; Level 5
+                   pwmint,         ; Level 6
+                   pwmint,         ; Level 7
+                   cmdint,         ; Level 8
+                   cmdint,         ; Level 9
+                   hint,           ; Level 10
+                   hint,           ; Level 11
+                   vint,           ; Level 12
+                   vint,           ; Level 13
+                   vresint,        ; Level 14
+                   vresint         ; Level 15

noret:
                rts
                nop

;-----------------------------------------------------------*
;       VRES Interrupt
;-----------------------------------------------------------*

vresint:
                mov.l       #_sysreg, r0
                ldc         r0, gbr

                move.w      r0, @ (vresintclr, gbr)     ; V Interrupt Clear

                mov.b       @ (dreqctl, gbr), r0        ; VRES corrective action
                tst         #RV, r0
                bf          vresloop

                mov.l       #S_STACK-8, r15             ; Stack change
                mov.l       #_hotstart, r0
                mov         r0, @r15                    ; PC change
                mov.w       #h f0, r0
                mov         r0, @ (4, r15)              ; SR mask

                mov.l       #_DMAOPERATION, r1
                mov         #0, r0
                mov.l       r0, @ r1                    ; DMA Off

                mov.l       #_DMACHANNEL0, r1
                mov         #0, r0
                mov.l       r0, @ r1
                mov.l       #b' 0100010011100000, r1
                mov.l       r0, @r1                     ; Channel Control

                rte
                nop
vresloop:
                bra         vresloop
```